Approaching a nano-satellite using CAN-SAT systems

Mohamed Abo-Arais[6], Mohamed Elfeki, Abdullah Mohammed, Mahmoud El-Mohr, Belal Abdelmoteleb, Ahmed Hatem, , Abo-Bakr Mohamed, Ahmed Ramy, Khaled Walid, Mona El-Fiky, Ahmed Darwish.
Faculty of Engineering, Alexandria University.
Alexandria, Egypt.

Mostafa Abd Al-Kareem, Walid El-Ajmi, Mohamed Gaber, Mohamed Salah, Ahmed Sobhy.

Alexandria Higher Institute of Engineering and Technology (AIET).
Alexandria, Egypt.

***Abstract*—Conquest of the space spreads with small educational projects that aim to explore the capabilities and the potentials that the mankind have. This paper concludes our experiments that led to launching the second Can-Sat form Alexandria, Egypt under affiliation of Space and Technology Program in Bibliotecha Alexandrina. Here we presents our satellite system and the communication system that connects the ground station to the flying system, as well it represents the launching procedure, and some of the problems we overboard during our project duration**

Keywords——satellite systems, dead-reckoning, sensor fusion, embedded systems, nano-satellites.

I. INTRODUCTION

Although the science of Space technology was monopolized many years by giant science companies and agencies due to the high tech requirements for such kind of technology, lately it has been spread to be within the reach for undergraduate students in multiple engineering fields, and even high school students to participate in the achievements happening in that field. In Alexandria, Egypt and under affiliation of Bibliotecha Alexandrina, Space and Technology program, we -Some second and third year engineering students- were capable of launching our first semi cube-sat to the space and retrieving it using parachute recovery system that was controlled automatically by Gyroscope and Accelerometer sensors to eject in the right time. However the autonomous control system that we equipped our satellite system with, we also made some emergency protocols to avoid any unexpected events to occur. In the following paper, we introduce to you our system that been launched using a water rocket to simulate the real satellites that been launched out of the earth through decades of theories and experiments in space technology.

II. BACKGROUND

A. The Concept of the Can-Sat

The can-sat concept was first introduced in the late 1990's by Robert J. Twiggs at Stanford University initiating the idea of what became the nano-satellite projects afterwards[1]. The idea is to launch a structure of the size of a soda can into the space. Its volume should be around 350 ml and the mass around 500 grams[2]. Later, the idea was evolved by Jordi Puig-Suari of California Polytechnic State University and Robert J. Twiggs to enable graduate students to be able to design, build, test and operate in space a spacecraft with capabilities similar to that of the first spacecraft, Sputnik which was the first identification of the Cube-Sat concept. This concept as initially proposed did not set out to become a standard; rather, it became a standard over time by a process of emergence [3].

B. Bringing Nano-Satellite Projects into real world

In 1999, a project called ARLISS involving mostly American and Japanese Universities, carrying out the first launch of a nano-satellite system, continuing each year without interruption. The rocket used was capable of moving 1.8 kilos and of ascending to 4000 meters, opening the door to low cost space flights -about $400[2]. Later, the tasks gone too different, for instance: calculating the opening of a landing system using data provided by the barometer or making use of differential GPS system.

III. OUR SYSTEM

Our satellite system consists of three fundamental subsystems: Ground-Station, Communication System and Satellite body. The ground-station is a desktop application that manipulate and control the satellite on real-time basis using the communication system. The communication system by itself is divided into two subsystems: Zigbee protocol and 3G World-wide web network. The last subsystem is the satellite body and it consists of two controllers: Arduino and Raspberry Pi v.2 and humidity, temperature, pressure, GPS sensors beside three web-cams used to live-stream the videos and captures while running-time.

For launching the satellite we used a water rocket system to enable reasonable height with very limited resources, so it was considered as an independent subsystem by its own.

Figure 1 represents a summarized overview of our system architecture and fundamental components, this block diagram describe the interfacing between components and the objective of each of them.

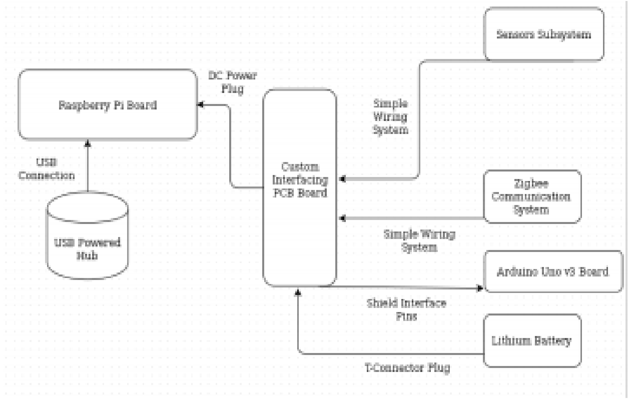
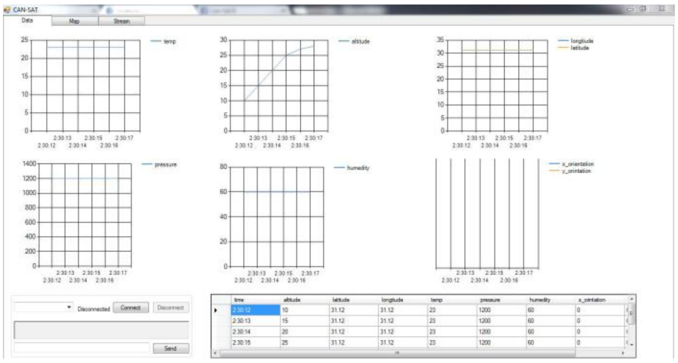


Figure 1: Satellite system architecture overview

The aim of this project was to build a nano-satellite has the ability to collect data from environmental services and prediction against natural changes. Measuring and monitoring should be live and internet based streaming to allow collection of data from world-wide and device independent. In the following section we will discuss each subsystem individually as follows:

A. Ground Station and Communication System

The Ground-Station was a desktop application written in C# to control, manipulate the data sent by the satellite while being on the air, it also had a data acquisition system, that allow plotting the data in graphs to clearly describe the changes while on run-time. Figure 2 shows our application while launching the satellite. This desktop application was connected to the satellite using two protocols: Ad-hoc network using (IEEE 802.15) Protocol and 3G Gcom®.



time	altitude	latitude	longitude	temp	pressure	humidity	x_orientation
2:30:12	10	31.12	31.12	23	1200	60	0
2:30:13	15	31.12	31.12	23	1200	60	0
2:30:14	20	31.12	31.12	23	1200	60	0
2:30:15	25	31.12	31.12	23	1200	60	0

Figure 2: Ground Station Data Acquisition System.

In order to be able to stream a live video, we first need the presence of two main specifications in our communication link; a large bandwidth, and a fast data rate transmission in both hardware and software settings. Instead of trying to solve these problems using the used Atmega AVR micro-controller, we used an independent micro-processor board that was able to launch a Linux distribution that was also supported by other hardware component like web-cams. So, we used Raspberry Pi model B development board with a 700~800 MHz microprocessor, 512 MB RAM and a modest VGA settings. Having these features, we were able to install “avconf” Linux library which is the updated version of “ffmpeg” that was used to record, taking static pictures, and streaming videos on-line. We also needed a portable internet connection provider that may be used on the satellite body, so that we used the 3G USB modem on the Raspberry Pi board. However, the 3G is not automatically recognized on the embedded linux kernel, so that we had to either write a device driver for it from scratch, or to find a suitable pre-written driver. The driver used was “sakis3g” tool, which was perfectly suitable for our needs. We also installed two additional digital cameras; one for taking snapshots and storing them into the SD card storage, and the other camera is for recording videos and storing them into the SD card storage.

In order to take snapshots, we wrote a Linux bash script that stores the command used to take a single static picture frame into a loop that whenever the camera connection restarts or crashes is able to restore its state and regain the ability to take static pictures in case of emergencies. The same idea was used for video recording, with other bash script. So, when any of the cameras restarts or crashes, that does not mean that the old stored data will be overwritten with the new one, but instead each stored piece ODF data will still exist even after a couple of crashes and reboots.

In order to automate the whole process, we have modified the login Linux bash script “.bashrc”, which is the first to be executed after the creation of new tty terminal in Linux kernel, so we appended the 3G driver we are using to automatically connect to internet via the 3G USB modem attached to the Raspberry Pi. We also invoked the three bash scripts that execute each of the following: live-streaming, storing snapshots, and recording videos on SD card. We executed all of the commands in background in order not to delay the execution of any subsystem in case of error interruption in other subsystem to ensure the integrity and robust of our model.

B. Satellite System and Sensor Control

1. Gyroscope

A gyroscope is a device that measures angular velocity, In our system we need to calculate the leaning angle to detect whenever the system become critically stable or in the worst case unstable so, it can -with the accelerometer- eject the parachute system in the perfect time. In order to obtain angles, integration over time is needed which makes some drifts due to the approximations made to integrate with micro-controllers and the fact of any error will be accumulated over time, that’s why using a gyroscope for angles calculation was not sufficient alone.

IDG500 Breakout Gyroscope was used, it’s a two axes gyroscope with analog output with a sensitivity of 2mV/s and a reference of 1.35 V which means that if the angular velocity = zero the output would be 1.35V. Notice that this value (1.35) is changeable due to ambient temperature so it’s not a fixed value to initialize.

2. Accelerometer

The accelerometer is a device that measures the proper acceleration -including the gravitational acceleration-. ADXL335 accelerometer was used, it’s a 3 axes accelerometer with analog output with a sensitivity of 300mV/g. For the gyroscope it has drifts over time due to integration but it is still a reliable short term way, however, for accelerometer it’s sensitive to noise and not hundred percent reliable if not in static state but it’s more reliable than gyroscope in long term as its drifts is not based on time. So we found that sensor fusion is the best suitable option to output the most accurate measurements over the individual sensor usage. The sensor fusion is to apply a low pass filter into both sensors and via a simple approach you can develop a decision making algorithm to eject the parachute in the perfect time. The complementary filter applied equation is as follows:

$$\Theta_f = (\Theta_f + \omega * \Delta t) * 0.9 + \Theta_A * 0.1 \qquad (1)$$

Where Θ_f is the filtered angel, Θ_A is the angel from accelerometer, ω is the angular velocity, and Δt is the difference in time.

3. Barometer

BMP is a sensor that measures the pressure and temperature using BMP085 module via I2C communication protocol. Unfortunately Pressure measurement were noisy and unstable because of the low quality sensor we used due the lack of resources, so we applied a low pass filter to achieve some sense of accuracy and reliability on measurements.

Moreover, the altitude may be calculated from pressure using the following formula:

$$\text{altitude} = 44330 * (1 – (p/p_0)^{1/5.255}) \qquad (2)$$

4. GPS

A GPS is a device that receives Global Positioning System satellites' signals to determine the device's location on Earth. GPS devices provide latitude and longitude information, and some may also calculate altitude and regional time. We used here Sky Labs SKM53 GPS module.

5. Humidity

RHT03 is a sensor that measures relative humidity as a percentage and temperature, using 1 wire protocol.

C. Launching Subsystem

The launching procedures for can-sat systems usually are executed using one of the following models: Rocket model, RC plane model, Balloon model, dropping from high raised building. In our case we chose the best model which is the first by default, however due to the lack of resources we could not afford launching a chemical rocket so we used the alternative water rocket model as our approach.

1. Theory of Operation

This rocket type uses water as its reaction mass, the pressure vessel (rocket engine) is usually a plastic soft drink bottle. The water is forced out by a pressurized gas, typically compressed air. To achieve high launching altitude, there are multiple approaches that use the water rocket model: Multi-bottle Single stage rockets, Multi-stage rocket, and single stage single bottle rocket.

After several experiments, we decided to work on single bottle model, as the multi-stage and multi-bottle approaches seemed to need additional resources to success that we already did not have. For example, in the following figure 3, the multi-bottle rocket can be unreliable, as according to the sensitivity of the initial conditions assumed in Lorenz Theorem, any tiny failure in sealing the rocket can cause the different parts to separate, which may off course cause the rocket to veer.

Due to the lack of resources we already abandoned multi-bottle and multi-stage designs, however that did not stop us from trying to approach the best possible output with the single-bottle system. So through many experiments we reached the following results that could imply at any 2 litre soft plastic bottle can be used as a water rocket:

- The percentage of water inside the bottle, ranging from 30% to 40% of the size of the bottle.
- The compression ratio inside the bottle, 7-8 bar.
- The use of mixed salt and water, increases the altitude. − The use of mixed Soap and water, increases the duration of thrust.

Figure 3: Multi-Bottle Rocket System

2. Aerodynamics

The forces that act on the rocket as shown on the figure 4 are as follows: thrust, drag, lift and weight [5].

Thrust: is created by the compressed air, opposing the drag.

Drag: imagine sticking your hand out the window of a moving car, the force that pushes your hand back is the drag, and it works on slowing the rocket down when the thrust quits.

Lift: according to Newton's laws and Bernoulli's principal, the lift force is proportional to the square of the velocity as the rocket moves.

Weight: rockets with less weight, requires less thrust.

Center of Gravity and Center of Pressure: Every stable aerodynamic object should have the center of gravity above the center of pressure as shown in the figure 5.

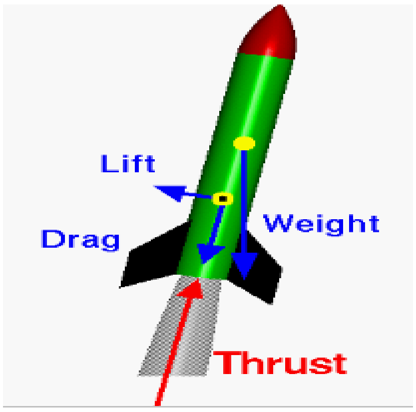


Figure 4: Forces applied on rocket

3. Parachute Function and Design

All model rockets require a recovery system to slow their descent and return them safely to the ground. The most common type of recovery system is the parachute. The parachute may be made from thin plastic or cloth. The parachute is expelled from the body tube by the ejection charge of the rocket motor after a delay to allow the rocket to reach apogee and be traveling at a relatively slow speed.

The key design parameters usually are the drag Coefficient , area and the suitable design. The area can be estimated from the following equation:

$$A_p = 2R / (\rho C_d V^2) \qquad (3)$$

Where A_p is the area of the parachute, ρ is the density of material, V descending velocity, C_d drag coefficient.

The parachute is placed in the top volume of the rocket with the servo motor and spring to push the parachute out the rocket to be free to open. The servo motor take the signal from the system based on accelerometer and gyroscope measurements to turn over and release the rubber band that frees the cone to release the parachute as shown in the figure 5.

The parachute is placed in the top volume of the rocket with the servo motor and spring to push the parachute out the rocket to be free to open. The servo motor take the signal from the system based on accelerometer and gyroscope measurements to turn over and release the rubber band that frees the cone to release the parachute as shown in the figure 6.

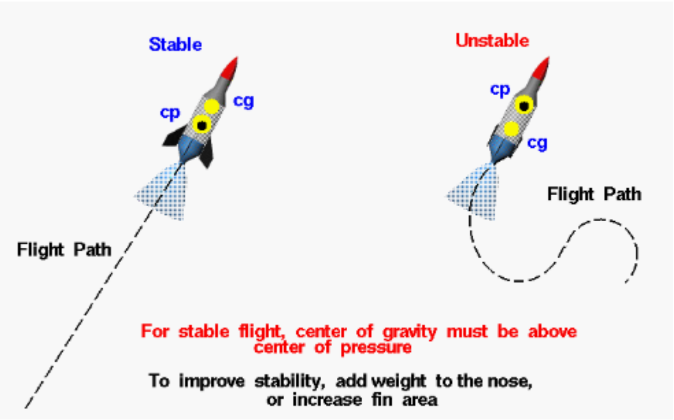


Figure 5: Stability due to center of gravity and pressure.

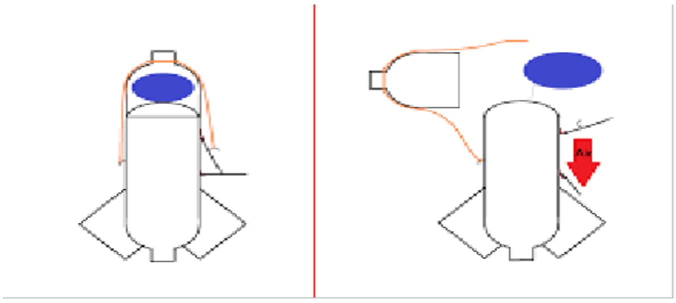


Figure 6: Parachuting System

IV. CONCLUSION

After the text edit has been completed, the paper is ready for the template. Duplicate the template file by using the Save As command, and use the naming convention prescribed by your conference for the name of your paper. In this newly created file, highlight all of the contents and import your prepared text file. You are now ready to style your paper; use the scroll down window on the left of the MS Word Formatting toolbar.

A. Authors and Affiliations

In this paper, we presented our satellite system that composed of three fundamental subsystems: Ground-station, Communication System, and satellite body. The groundstation was a C# desktop application responsible for controlling the satellite and representing the data received live from the satellite and plotting the data in graphs to make the changes clear to the observer. The communication system was a multi-component system consists from two main components: Zigbee communication Protocol and livestreaming through world-wide-web network and 3G coverage embedded into the satellite. And the last subsystem was the satellite body holding two controllers: Arduino and RasbperryPi v.2 reading measurements from humidity, temperature, pressure, GPS, and three web cameras recording and streaming videos, images, and sensors measurement to the ground station via the communication protocols.

The Launching was implemented using the water rocket single bottle launching model due to lack of resources, however we believe in Neil Armstrong saying, that “A small step for a man, a giant leap for a mankind”.

Figure 7: Can-Sat after parachute ejection

Acknowledgment

All of our code and documentations are pushed as open-source resources on [4].

Also we would like to thank: Hazem K. Abd latif, Yassmin Hamdy, Zeyad Mamdouh and Khalid Gamal, because they helped a lot with the procedures and experiments while project duration.

Author Mohamed Abo-arais[6] is also affiliated by Ain-Shams University, Cairo, Egypt.

References

[1] Robert J. Twiggs, / cansatws / programandabstract.pdf "Introducing New Challenges for Future Space Missions", International CanSat Workshop, February 23, 2007.

[2] R. Walker et al., "ESA Hands-on Space Education Project Activities for University Students: Attracting and Training the Next Generation of Engineers Space ", 14–16 April 2010.

[3] Helvajian2008, Henry; editors, Siegfried W. Janson, (2008). Small Satellites: Past, Present, and Future. El Segundo, Calif.: Aerospace Press. Transl. J. Magn. Japan, vol. 2, pp. 740-741, August 1987 [Digests 9th Annual Conf. Magnetics Japan, p. 301, 1982].

[4] Bitbucket.org,. 'Can-Sat-2014 / Can-Sat-2015 / Source / — Bitbucket'. N.p., 2015. Web. 27 Feb. 2015

[5] Exploration.grc.nasa.gov,.'Rocket Thrust'. N.p.,2015. Web. 3 Mar. 2015.